What is claimed is:

1 1.   A method of identifying reusable computation units comprising:

2   mapping n-dimensional architectural state vectors into a plurality of one-

3 dimensional symbols;

4   arranging the plurality of one-dimensional symbols into phrases of text; and

5   identifying recurrent phrases of text as reusable computation units.

1 2.   The method of claim 1 wherein mapping comprises:

2   traversing a software block in program execution order;

3   assigning new symbols as previously un-encountered architectural state

4 vectors are encountered; and

5   assigning previously assigned symbols as previously encountered

6 architectural state vectors are encountered.

1 3.   The method of claim 2 wherein assigning new symbols comprises assigning

2 consecutive integers such that each new symbol is assigned a value that is one

3 greater than a previously assigned value.

1 4.   The method of claim 1 wherein arranging comprises arranging symbols in

2 program execution order.

1 5.   The method of claim 4 wherein architectural state vectors include live-in

2 states and live-out states for individual processor instructions.

1 6.   The method of claim 1 wherein identifying comprises compressing the

2 phrases of text to find a plurality of recurrent phrases.

1 7.   The method of claim 6 wherein compressing comprises compressing the

2 phrases of text using a lossless compression algorithm.

1 8.   The method of claim 7 further comprising generating at least one trigger for a

2 conjugate processor, the at least one trigger to implement complete reuse.

1 9.   The method of claim 6 wherein compressing comprises compressing the

2 phrases of text using a lossy algorithm.

1 10.   The method of claim 9 further comprising generating at least one trigger for a

2 conjugate processor, the at least one trigger to implement partial reuse.

1 11.   The method of claim 6 wherein identifying further comprises correlating the

2 plurality of recurrent phrases to identify reusable computation units.

1 12.   The method of claim 1 further comprising annotating the reusable

2 computation units in a program binary to cause a processor to memorize reuse

3 instances.

1 13.   A computer-implemented method of identifying reusable computation units

2 within an executable program comprising:

3   creating an execution trace of the executable program;

4   compressing the execution trace to find recurrent portions thereof; and

5   identifying the recurrent portions of the execution trace as reusable

6 computation units.

1 14.   The computer-implemented method of claim 13 wherein creating an

2 execution trace comprises:

3   executing the executable program; and

4   mapping architectural states of the executable program into symbols.

1 15.   The computer-implemented method of claim 14 wherein mapping
2 architectural states into symbols comprises:
3   assigning integers to n-dimensional architectural state vectors such that each
4 new n-dimensional architectural state vector is assigned an integer that is one greater
5 than the last integer assigned.

1 16.   The computer-implemented method of claim 15 wherein the n-dimensional
2 architectural state vectors include information from processor instructions, live-in
3 states, and live-out states.

1 17.   The computer-implemented method of claim 13 wherein compressing
2 comprises:
3   applying a compression algorithm that identifies an editing distance between
4 similar recurrent phrases.

1 18.   The computer-implemented method of claim 15 wherein identifying recurrent
2 portions of the execution trace comprises:
3   correlating the dictionary of recurrent phrases with the executable program.

1 19.   The computer-implemented method of claim 13 wherein compressing
2 comprises:
3   applying a compression algorithm that identifies a dictionary of recurrent
4 phrases.

1 20.   The computer-implemented method of claim 19 wherein identifying recurrent
2 portions of the execution trace comprises:
3   correlating the dictionary of recurrent phrases with the executable program.

1 21.   The computer-implemented method of claim 20 further comprising
2 annotating the reusable computation units in the executable program.

1   22.   The computer-implemented method of claim 20 further comprising

2   generating conjugate processor triggers to exploit reusable computation units in the

3   executable program.

1   23.   An article having a computer-readable medium, the computer-readable

2   medium having stored thereon instructions for a method of identifying reusable

3   computation units, the method comprising:

4   compressing phrases of symbols that represent architectural states to identify

5   recurrent phrases of symbols; and

6   correlating recurrent phrases of symbols with an executable program to

7   identify reusable computation units within the executable program.

1   24.   The article of claim 23 further comprising:

2   generating the phrases of symbols by mapping n-dimensional architectural

3   states to one-dimensional symbols.

1   25.   The article of claim 24 wherein mapping comprises:

2   executing an executable program; and

3   assigning an integer to each unique n-dimensional architectural state vector

4   representing a processor instruction, live-in states, and live-out states.

1   26.   The article of claim 25 wherein executing the executable program comprises

2   generating a program trace that includes the n-dimensional architectural state vectors.

1   27.   The article of claim 23 wherein compressing phrases of symbols comprises

2   applying a lossless coding algorithm to the phrases of symbols.

1   28.   The article of claim 27 wherein the method further comprises generating

2   instruction triggers for a conjugate processor to implement complete reuse.

1　29.　The article of claim 23 wherein compressing phrases of symbols comprises

2　applying a compression algorithm that identifies an editing distance between similar

3　phrases of symbols.

1　30.　The article of claim 29 wherein the method further comprises generating

2　instruction triggers for a conjugate processor to implement partial reuse.